Homework 1

EECS 270 Winter 2020

Due Friday, January 24 @ 11:59 PM on Gradescope

This is an individual assignment, all of the work should be your own.
Write neatly or type and show all your work for full credit.
Have your name and unique name on the front page of your submission.

1. **[20 points]** Convert the following decimal numbers to binary, and then to octal and hexadecimal. If the fraction requires more than 6 bits, truncate the fraction to 6 bits.

   a. 128

   b. 42555

   c. 681.875

   d. 13.37

   e. 808354.808354

Note that fractions are to be truncated to 6 **bits**: throw out all bits after the first 6 in the fraction before converting to octal and hexadecimal. Spacing for binary, octal, and hexadecimal representations is done for readability.

	dec	bin	oct	hex
a.	128	1000 0000	200	80
b.	42555	1010 0110 0011 1011	123 073	A63B
c.	681.875	10 1010 1001.111	1 251.3	2A9.E
d.	13.37	1101.0101 11	15.23	D.5C
e.	808354.808354	1100 0101 0101 1010 0010.1100 11	3 052 642.63	C 55A2.CC

2. **[16 points]** Convert the following binary numbers to decimal, octal, and hexadecimal.

   a. 1010

   b. 11101

   c. 100.11

   d. 10101111.0011

	bin	dec	oct	hex
a.	1010	10	12	A
b.	11101	29	35	1D
c.	100.11	4.75	4.6	4.C
d.	10101111.0011	175.1875	257.16	AF.3

3. **[12 points]** Convert the following hexadecimal numbers to decimal, binary, and octal.

   a. DA

   b. C4.E

   c. 42.58

	hex	dec	bin	oct
a.	DA	218	1101 1010	332
b.	C4.E	196.875	1100 0100.111	304.70
c.	42.58	66.34375	100 0010.0101 1	102.26

4. **[8 points]** Your coworker has handed you a binary dump of a file. The numbers on the left of the pipes indicate which position the left-most 8-bit datum on the row is, 0-indexed (e.g. on the top row, 01001000 is at position 0 and 01101100 is at position 3). Convert the 8-bit data into ASCII text (you should get actual words; write "NUL" when you encounter a 00000000):

```
0  | 01001000 01100101 01111001 00101100
4  | 00100000 01001001 00100000 01110100
8  | 01101000 01101001 01101110 01101011
12 | 00100000 01000101 01000101 01000011
16 | 01010011 00100000 00110010 00110111
20 | 00110000 00100000 01101001 01110011
24 | 00100000 01000110 01010101 01001110
28 | 00100001 00000000
```

```
0  |  'H'    'e'    'y'    ','
4  |  ' '    'I'    ' '    't'
8  |  'h'    'i'    'n'    'k'
12 |  ' '    'E'    'E'    'C'
16 |  'S'    ' '    '2'    '7'
20 |  '0'    ' '    'i'    's'
24 |  ' '    'F'    'U'    'N'
28 |  '!'    '\0'
```

Hey, I think EECS 270 is FUN!NUL
"Hey, I think EECS 270 is FUN!"

5. **[8 points]** Assume that a signal is encoded using 12 bits. Assume that many of the encodings turn out to be either 000000000000, 000001100000, or 111111111111. We thus decide to create compressed encodings by representing 000000000000 as 00, 000001100000 as 01, and 111111111111 as 10. 11 means that an uncompressed encoding follows. Using this encoding scheme, decompress the following encoded stream. Have a noticeable separator for each of the 12 bit signals:

```
00 00 10 11 000000000001 01 11 101100111001 00 10
```

```
000000000000
000000000000
111111111111
000000000001
000001100000
101100111001
000000000000
111111111111
```

6. **[20 points]** Determine the number of digits necessary to represent the following decimal numbers in binary, octal, decimal, and hexadecimal representations. For example, decimal 12 requires 4 digits in binary (1100), 2 digits in octal (14), 2 digits in decimal (12), and one digit in hexadecimal (C).

a. 05
b. 46
c. 256
d. 5,912
e. 65,537

The equation $\lceil log_b x \rceil$ determines how many digits of base b are needed to encode for x different things, in the case for this problem, how many unsigned numbers including 0. For example, to represent 8 different numbers, numbers 0 to 7, you would need $\lceil log_2 8 \rceil = 3$ bits.

Note that x is **not** the number you want to represent, but the quantity of different things you want to represent. If you wanted to represent the decimal number 8 in binary, you would need an additional bit for 4 bits, as 3 bits encodes for numbers 0 to 7. To compensate, you can add 1 to x in the equation, since to represent number 8, you technically need to encode for *9* things (when we introduce signed numbers, the method to compensate will change).

(Fun fact: this equation is so useful that Verilog 2005 specified a function called `$clog2()` that would calculate $\lceil log_2 x \rceil$. An important use is to figure out how many bits would be needed to represent memories or values that had a known maximum size.)

		bin	oct	dec	hex
a.	05	3	1	1	1
b.	46	6	2	2	2
c.	256	9	3	3	3
d.	5,912	13	5	4	4
e.	65,537	17	6	5	5

7. **[16 points]** Determine the decimal number ranges that can be represented in d digits for binary, octal, decimal, and hexadecimal numbers. For example, for $d = 3$, the binary range is [000, 111] which is [0, 7] in decimal. Similarly, the 3-digit hexadecimal range is [000, FFF] which is [0, 4095] in decimal.

   a. 1
   b. 2
   c. 4
   d. 8

With b being the base and d being the number of digits, b^d is the number of things that having d digits of base b can encode for. The equation $\lceil log_b x \rceil$ is derived from this. This is because each additional digit allows for more permutations: for each additional digit you get b times more permuations.

So far we have only been looking at unsigned/positive numbers. Our lower bound will be 0. For a given b^d encodings, since we have to represent 0, our upper bound ends up becoming $b^d - 1$. Thus, our number range will be [0, $b^d - 1$].

	digits	bin	oct	dec	hex
a.	1	[0,1]	[0,7]	[0,9]	[0,15]
b.	2	[0,3]	[0,63]	[0,99]	[0,255]
c.	4	[0,15]	[0,4095]	[0,9999]	[0,65535]
d.	8	[0,255]	[0,16777215]	[0,99999999]	[0,4294967295]